REȚELE NEURONALE ARTIFICIALE ÎN MATLAB

PERCEPTRONUL MULTISTRAT

Reţeaua neuronală de tip perceptron multistrat este construită foarte simplu în Matlab, folosind toolbox-ul Neural Networks.

Pentru construirea unei reţele, este necesară în prealabil, definirea perechilor de date de intrare-ieşire, în funcţie de care se vor dimensiona automat straturile de intrare şi de ieşire ale reţelei. Datele trebuie prezentate sub formă de matrice, în care numărul de linii îl reprezintă dimensiunea unui model de intrare (numărul de valori P Şi Q care definesc un regim de încărcare al reţelei) respectiv de ieşire (tensiunile nodale din reţea).

Iniţializarea reţelei neuronale (notată generic cu variabila net) se realizează cu ajutorul unei singure linii de program:

net=feedforwardnet(hiddenSizes,trainFcn);

unde

hiddenSizes - numărul și dimensiunea straturilor ascunse, număr dacă reţeaua are un strat ascuns sau vector dacă reţeaua are mai multe straturi.

trainFcn - algoritmul ales pentru antrenarea rețelei.

Exemple:

* apelul net=feedforwardnet va crea o RNA perceptron cu un strat ascuns alcătuit din 10 neuroni, care folosește algoritmul de antrenare Levenberg-Marquardt (‘trainlm’), parametri impliciți, daca nu se specifica alte valori de către utilizator.

**Atenție!** Algoritmul Levenberg- Marquardt este foarte performant, însă mare consumator de memorie. Ca alternativă, se recomandă algoritmul standard RProp (Resilient Backpropagation) – ‘trainrp’. Lista completă a algoritmelor de antrenare disponibile în toolboxul Neural Networks din Matlab poate fi consultată în manualul acestui toolbox, atașat acestui tutorial.

* net=feedforwardnet(20) va crea un perceptron cu un strat ascuns alcătuit din 20 de neuroni și care folosește algoritmul de antrenare Levenberg-Marquardt.
* net=feedforwardnet([20,10],’trainrp’) va crea un perceptron cu două straturi ascunse, primul cu 20 de neuroni şi al doilea cu 10 neuroni, care folosește algoritmul de antrenare Resilient Backpropagation.

Implicit, funcţia de activare a neuronilor de pe straturile ascunse este tangent hiperbolic, iar antrenarea reţelei se realizează pe parcursul a 1000 de cicluri. Aceste opţiuni, alături de multe altele de configurare avansată, pot fi modificate de către utilizator. Toate acestea sunt explicate în detaliu în manualul atașat acestui tutorial.

Antrenarea reţelei neuronale este comandată tot printr-o singură linie de program:

[net,tr] = train(net,intrări,ieşiri);

în care

* net este reţeaua neuronală antrenată
* intrări şi ieşiri sunt matricele de date care formează perechile de intrare-ieşire pentru care se realizează antrenarea.

Generalizarea sau folosirea reţelei pentru rezolvarea ulterioară a problemei se realizează tot cu o singură linie de program:

iesire=net(intrare);

unde

iesire – valoarea calculată de RNA, un vector care are un număr de elemente egal cu numărul de variabile de ieşire al problemei rezolvate.

intrare – noile valori de intrare pentru care se doreşte rezolvarea problemei, vector sau un număr.

**Exemplu numeric.**

Aproximarea funcției f(x)=2x2+4 folosind RNA şi toolboxul Neural Networks din Matlab.

**Notă!** În acest tutorial s-a folosit Matlab versiunea 2012a. Este necesară cel puţin versiunea 2010b.

1. **Pregătirea datelor folosite pentru antrenare.**

RNA de tip perceptron multistrat învață perechi de intrare-ieșire. Astfel, pentru a putea aproxima funcția de gradul 2 indicată, ea trebuie antrenată cu perechi x-f(x) de tipul celor din tabelul din Fig. 1. Apoi, dacă aceste perechi au fost alese relevant(1) și cuprinzător(2), rețeaua ar trebui să recunoască orice valoare a funcției, din intervalul considerat pentru variabilele de intrare. Astfel, pentru exemplul din Fig. 1, RNA ar trebui antrenată cu valorile **albastre** din tabel, cu coloana x pentru intrări şi coloana f(x) pentru ieşiri. După antrenare, RNA trebuie să fie capabilă să recunoască toate valorile f(x) din intervalul [-20,22], de exemplu **f(14)=396**.

|  |  |
| --- | --- |
| x | f(x)=2\*x2+4 |
| **-20** | **804** |
| **-17** | **582** |
| **-14** | **396** |
| **-11** | **246** |
| **-8** | **132** |
| **-5** | **54** |
| **-2** | **12** |
| **1** | **6** |
| **4** | **36** |
| **7** | **102** |
| **10** | **204** |
| **13** | **342** |
| **16** | **516** |
| **19** | **726** |
| **22** | **972** |
| **14** | **396** |

Fig. 1 – Exemplu pentru antrenarea unei RNA

1. relevanța se referă la furnizarea corespunzătoare a datelor de intrare, ținând cont de utilizarea ulterioară a RNA în rezolvarea problemei. Pentru exemplul de mai sus, dacă s-ar indica perechi de învățare doar pentru x<0, când variația funcției f(x) este descrescătoare, ar fi probabil ca RNA să nu furnizeze rezultate corecte pentru zona crescătoare, când x>0, deoarece curba funcţiei nu ar fi descrisă în totalitate.

1. datele de învăţare trebuie să reflecte cât mai fidel variaţia ieşirilor în funcţie de intrări. De pildă, dacă din setul de antrenare al problemei indicate mai sus ar lipsi date din intervalul x=[-10,10], unde funcţia îşi are extremul, recunoaşterea ulterioară a valorilor din acest interval poate fi greşită. Aceasta este valabil mai ales în cazul funcţiilor complexe, care depind de mai multe variabile de intrare. De exemplu, dacă s-ar dori aproximarea sau prognoza consumului de energie electrică în funcţie de variaţia temperaturii şi a vitezei vântului, nu ar fi suficiente doar date pentru (vreme caldă-vânt slab) şi (vreme rece-vânt puternic), ci trebuie indicate şi date pentru (vreme caldă-vânt puternic) şi (vreme rece-vânt slab).

Avantajul folosirii RNA va fi faptul că, după antrenare, nu mai este necesară cunoaşterea expresiei analitice a funcţiei f(x), ea fiind stocată în ponderile şi pragurile reţelei. Aceasta reprezintă un avantaj major în două cazuri:

* atunci când nu se cunoaşte expresia funcţiei ce se doreşte a fi aproximată şi sunt disponibile doar tabele de date cu perechi de tip x-f(x);
* atunci când reţeaua neuronală înlocuieşte un algoritm complicat, care necesită multe calcule pentru obţinerea soluţiei căutate a problemei.

Datele pentru antrenare trebuie introduse sub forma unor de matrice, în care numărul de linii îl reprezintă dimensiunea unui model de intrare (numărul de variabile de intrare al problemei rezolvate, în cazul exemplului nostru 1, valoarea x) respectiv de ieşire (numărul de variabile de ieşire al problemei rezolvate, în cazul nostru 1, f(x)).

Datele de intrare, constând în valori x-f(x) pentru toate valorile pare ale lui x din intervalul [-100,100], au fost salvate in doua fişiere text, intrare.txt şi iesire.txt, de unde pot fi încărcate în Matlab folosind comenzile:

load intrare.txt -ascii;

load ieşire.txt -ascii;

Încărcarea fişierelor creează în Matlab variabilele omonime, ce vor fi folosite la antrenarea reţelei.

**Atenţie!** Pentru ca fişierele să poată fi încărcate, ele trebuie să se găsească în directorul curent de lucru!

1. **Definirea reţelei**

Se defineşte o RNA cu un strat ascuns alcătuit din 4 neuroni, care va folosi algoritmul de antrenare implicit, Levenberg-Marquardt. Numărul de neuroni de pe stratul de intrare şi de pe cel de ieşire va fi configurat automat, fără intervenţia utilizatorului, în funcţie de numărul de linii al matricelor intrare şi ieşire încărcate ca date de antrenare. În cazul exemplului nostru, ele vor avea câte un neuron, deci reţeaua completă va avea 1-4-1 neuroni.

net=feedforwardnet(4);

**Notă!** Daca se alege un număr prea mare sau prea mic de neuroni pe un strat, raportat la dimensiunile problemei rezolvate, antrenarea se poate opri prematur, iar rezultatele pot fi nesatisfăcătoare.

1. **Antrenarea reţelei**

[net,tr] = train(net,intrare,ieşire);

Procesul de antrenare poate fi urmărit în fereastra care se deschide automat după lansarea antrenării. Utilizatorul poate vedea, printre altele, structura reţelei (Neural Network), algoritmul de antrenare utilizat (Training), numărul de cicluri de antrenare efectuat (Epoch) şi evoluţia erorii pe setul de antrenare (Performance) (Fig. 2). Utilizatorul poate întrerupe oricând procesul de antrenare apăsând butonul **Stop Training** (opreşte antrenarea), iar la sfârşitul antrenării poate vizualiza grafic evoluţia erorii pe parcursul procesului de antrenare, apăsând butonul **Performance** (Fig. 3).

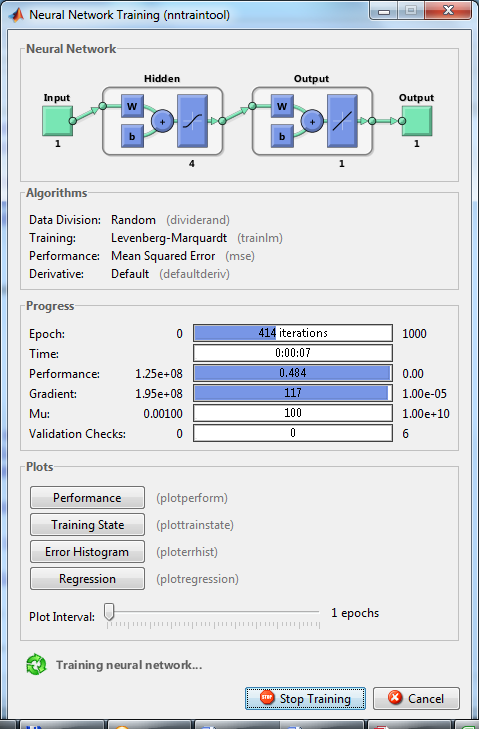


Fig. 2 - Antrenarea reţelei neuronale în Matlab

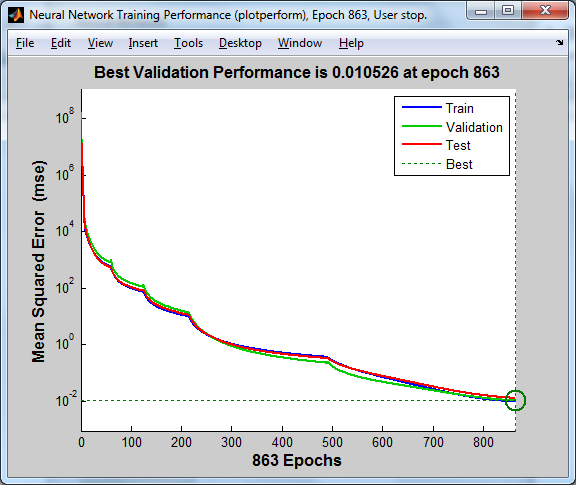


Fig. 3. **-**  Evoluţia erorii pe setul de antrenare

Un indiciu al unei antrenări corecte, dar nu întotdeauna relevant, este scăderea continuă a valorii parametrului Performance din Fig. 2, care se doreşte a fi cât mai mică (subunitară). În cazul exemplului prezentat, ea are, după 400 de iteraţii, o valoare de sub 0.5.

Cu cât eroarea de antrenare e mai mică, cu atât rezultatele ar trebui să fie, teoretic, mai precise.

1. **Recunoaşterea**

Reţeaua antrenată poate fi folosită ulterior pentru calculul valorilor funcţiei cu care s-a făcut antrenarea, pe intervalul considerat al valorilor de intrare. Valoarea obţinută nu va fi cea exactă, ci o aproximaţie apropiată. Eroarea soluţiei obţinute depinde de mai mulţi factori: structura aleasă pentru RNA (număr de straturi, număr de neuroni pe straturi, funcţiile de activare ale neuronilor), validitatea setului de date de antrenare (alegerea optimă a perechilor de învăţare x-f(x)), valorile iniţiale ale ponderilor şi pragurilor.

De pildă, dacă după antrenare se doreşte calculul valorii funcţiei pentru x=17.1, valoare neinclusă în setul de antrenare, se va obţine valoarea:

>> cc=net(17.1)

cc =

588.7153

Valoarea exactă a funcţiei este f(17.1)=588.82.

**Notă!** La reproducerea paşilor tutorialului, valoarea f(x) calculată de reţeaua neuronală va diferi de cea prezentată ca exemplu, deoarece antrenarea porneşte de la ponderi şi praguri ale neuronilor reţelei iniţializate aleatoriu în Matlab.

Salvarea reţelei şi a datelor folosite la antrenarea ei se poate face folosind comanda standard save din Matlab.

Încărcarea ulterioară a valorilor salvate se face cu comanda load, având grijă ca variabila încărcată să se găsească în directorul curent de lucru.

Pentru probleme mai complexe, este necesară definirea fişierelor de date conform cu specificul problemei şi alegerea corespunzătoare a structurii RNA. În rest, procedura de calcul rămâne la fel de simplă.